
**HAUTE ECOLE DE NAMUR-LIEGE-LUXEMBOURG**

## DEPARTEMENT INGENIEUR INDUSTRIEL

## PIERRARD - VIRTON

# SYSTEMES INTELLIGENTS

# ArcadeCube

Professeurs : R.Slama – C.Liberatore

Elèves : DEMIR Yasin – SCHYNS Pierre

Master 1 - Automatisation

Année académique

2018 -2019

# Table des matières

# I. Préambule

Dans le cadre du cours de systèmes intelligents, nous devons réaliser un projet dans lequel nous devons utiliser différentes technologies comme une carte Arduino et/ou un Raspberry. Pour ce projet, il faudra développer un système utilisant de l'intelligence artificielle.

## 1. Projet ArcadeCube


Notre projet consiste à développer et construire une borne de jeux arcades (Super Mario Bros 3) avec une reconnaissance gestuelle pour faciliter et/ou complémenter les joysticks.

Pour la réalisation de ce projet, nous utiliserons un Arduino, deux Raspberry Pi 3 et une caméra Pi. Les autres éléments seront détaillés plus loin, dans le chapitre « Matériel nécessaire ».

## 2. Objectif

En fin de projet, la borne arcade permettra de jouer à Super Mario Bros 3, les côtés du boitier comportera des LEDS qui s'allumeront en fonction de l'intensité de la musique du jeu. La caméra à une certaine hauteur, contrôlera et détectera les gestes de la main. Par exemple, un doigt levé permettra de mettre le jeu en pause.

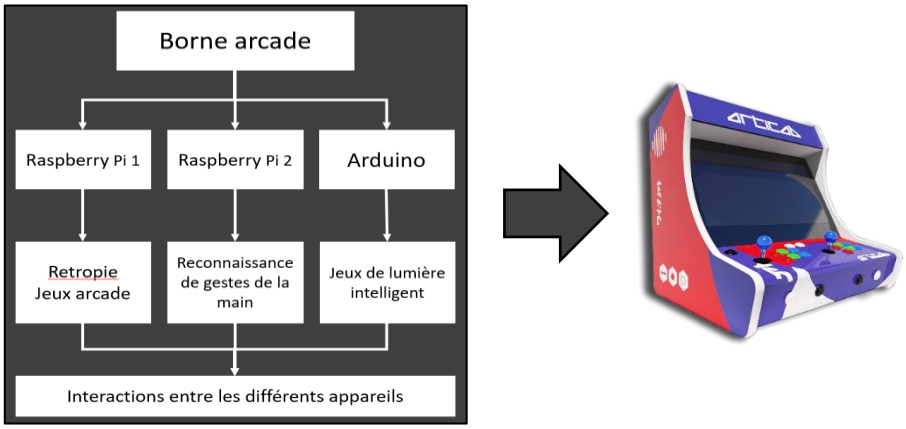



*Figure 1: Diagramme : fonction des appareils & objectif finale (exemple)*

Le Raspberry Pi 1 sera utilisé pour installer le système d'exploitation Retropie. Ce dernier combiné avec des joysticks permettra de jouer aux jeux arcades.

Retropie est un système d'exploitation permettant d'émuler des consoles de jeux arcades comme par exemple « Super Nintendo ».

Le Raspberry Pi 2 sera utilisé avec une caméra et un algorithme pour la reconnaissance gestuelle. Lorsque le doigt levé est détecté, le jeu se met en pause ou se retire du mode pause.

L'Arduino, comme dit ci-dessus, sera utilisé avec un capteur de son analogique pour afficher l'intensité de la musique. Plus la musique sera forte, plus il y aura de LEDS allumées.

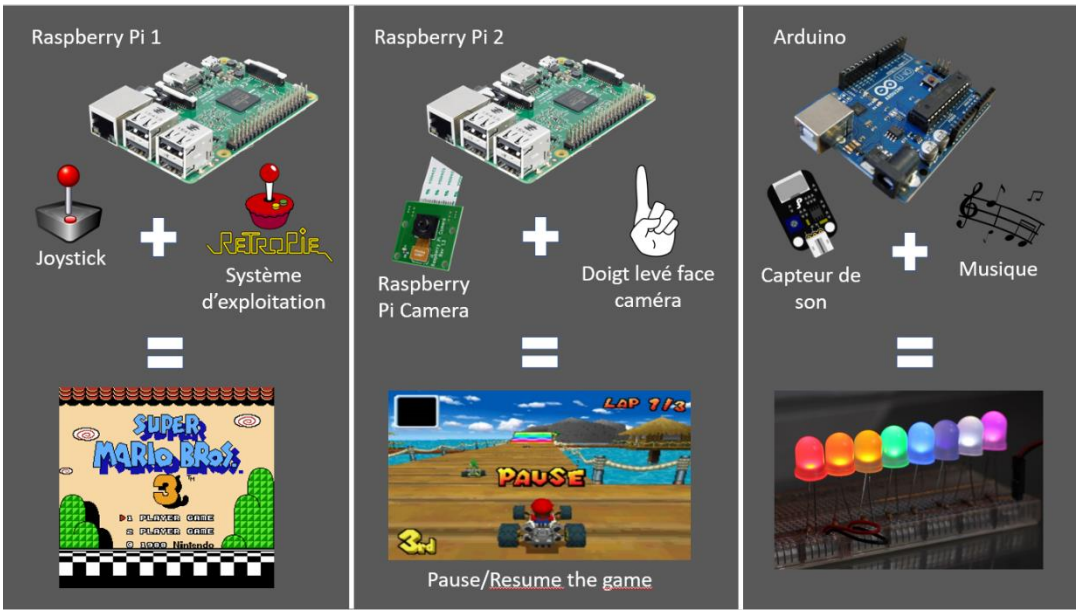



*Figure 3: Détail des différentes fonctions des appareils*

## Matériel nécessaire

*Tableau 1 : Liste du matériel nécessaire et leur prix*

| Nom du matériel | Prix | Quantité | Liens / Magasins | Fournit par l'école |
|---|---|---|---|---|
| Joysticks | 39.00€ | 1 | Amazon | Non |
| Raspberry Pi 3 Kit | 54.80€ | 2 | Amazon | Oui |
| Arduino Kit | 86.20€ | 1 | Amazon | Oui |
| Cable HDMI | 5.79€ | 1 | Amazon | Oui |
| HDMI / VGA | 7.49€ | 1 | Amazon | Oui |
| Moniteur avec haut-parleur intégré | 129.99€ | 1 | Amazon | Oui |
| Bois (borne) | 19.95€ | 1 | Hubo | Non |
| Vis | 11.32€ | 1 | Amazon | Non |
| Multiprise | 5.58€ | 1 | Amazon | Non |
| Capteur de son | 4.90€ | 1 | GoTronic | Non |
| **TOTAL** | **365.02€** | | | |

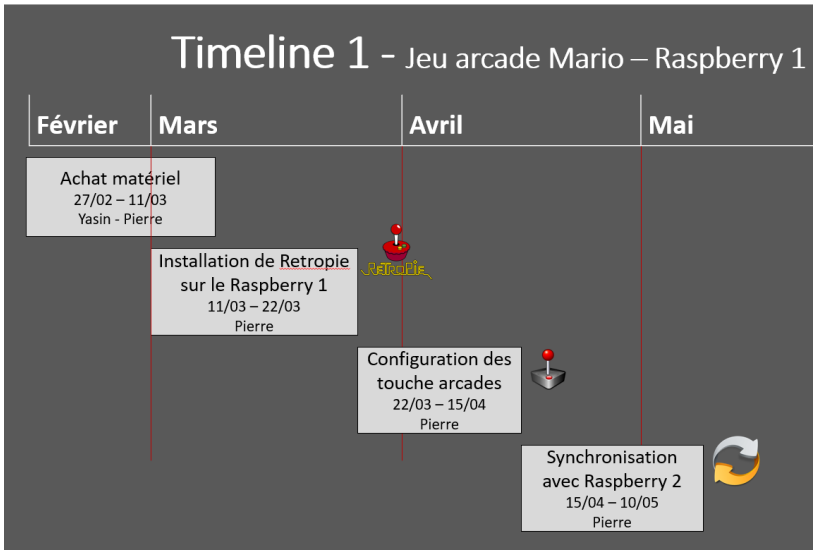
Des LEDS sont fournies dans la boîte de l'Arduino. L'école fournit également des LEDS pour le projet de lumière. Ce tableau ne prend pas en compte les matériaux de soudage nécessaire pour l'assemblement des différentes composants électroniques.

Pour la Raspberry 1, nous utiliserons le système d'exploitation Retropie. Les joysticks seront raccordés via un connecteur USB à la Raspberry (les câbles sont fournis avec les joysticks). Nous aurons besoin d'un écran pour visualiser les jeux. Le haut-parleur permettra de diffuser le son du jeu en cours.

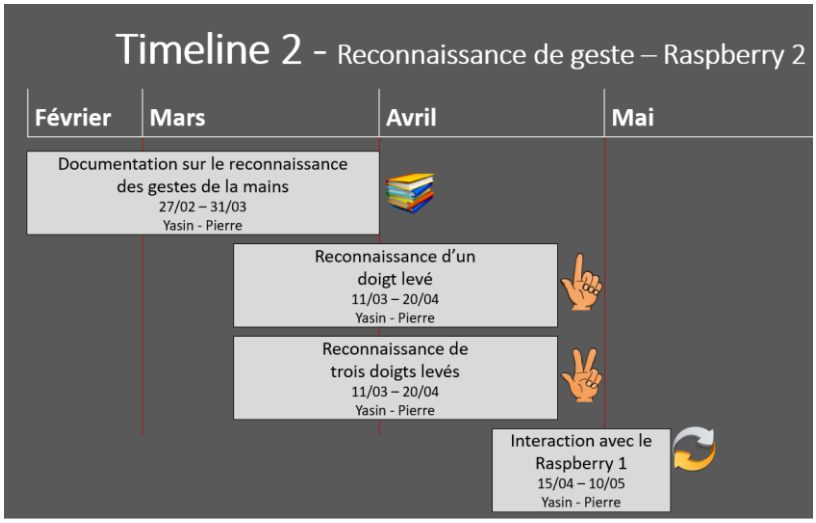
Pour la Raspberry 2, nous utiliserons la camera pour obtenir les images de la main. Des câbles et transistors seront nécessaire pour la communication entre les 2 Raspberry.

Pour l'Arduino, nous utiliserons un capteur de son afin de déterminer le nombre de LEDS qui seront allumées.

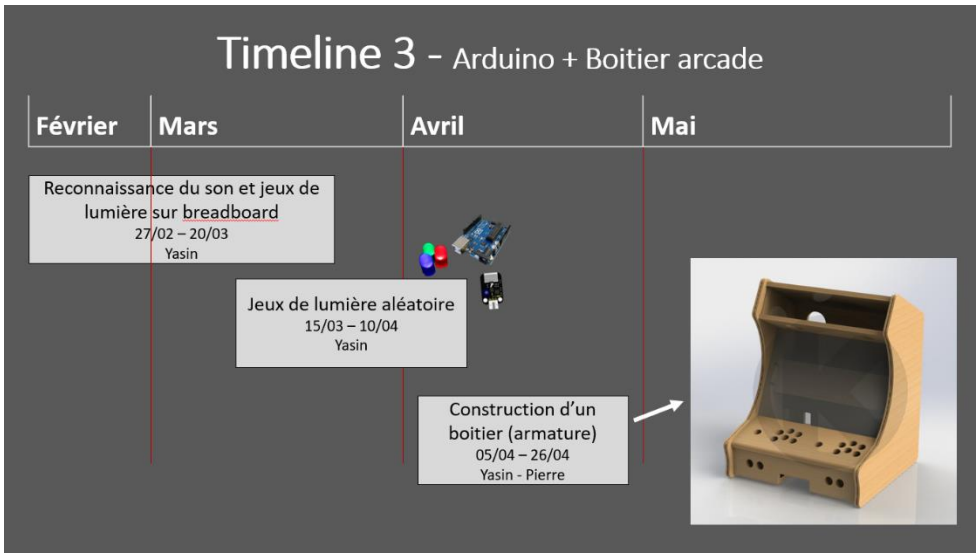
## 3. Diagramme du projet – Répartition des tâches




Mis à part le dernier point « Synchronisation avec Raspberry 2 », il n'y a pas eu de problème avec les échéances.




Les différents problèmes rencontrés lors de l'acquisition de connaissance et lors de tests de différents projets, à causer des dépassements au niveau des échéances. Cependant, la réalisation d'un programme pour la synchronisation avec Raspberry 1 a été un succès malgré le dépassement.




Les échéances ont été respectées.

## II. Partie Arduino

Cette partie de code permet d'allumer une LED sur quatre en fonction de l'intensité de la musique. Elle est issue du site internet « wikidebrouillard ».

Diagramme de fonctionnement [4] :

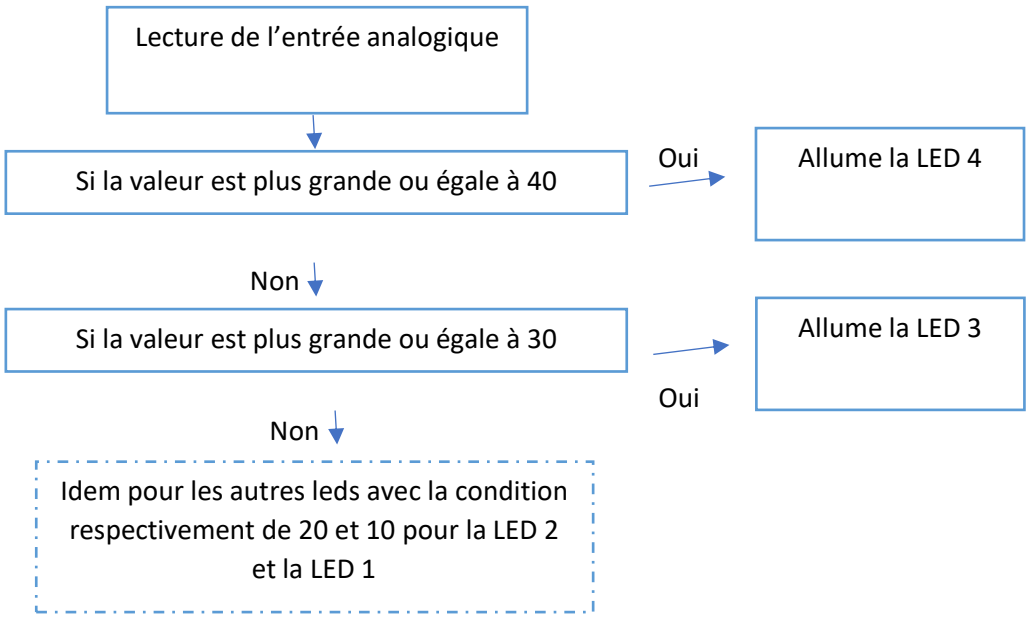



*Figure 4 : Diiagramme - fonctionnement*

La première partie du code permet de définir les sorties de l'Arduino. Le programme lit l'information analogique du capteur de son et en fonction de sa valeur, allume une des quatre LEDS. Voici la disposition des différents composants pour l'Arduino. [5]

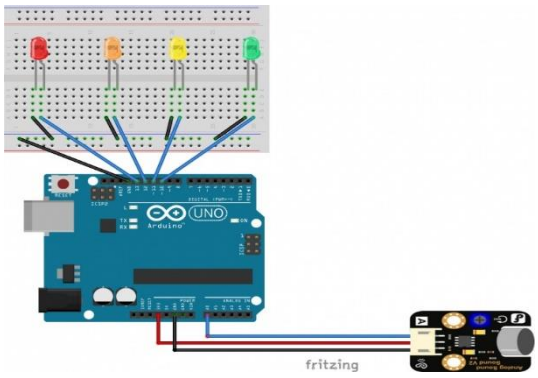



*Figure 5 : Circuit du modèle Wikidebrouillard*

L'analyse du programme, nous permet de réfléchir à des modifications et des améliorations (lien):

- Les LEDS doivent s'allumer en fonction de l'intensité de la musique (il ne faut pas qu'elle s'allume une par une) ;
- 4 LEDS ne suffisent pas pour le projet, il faut donc augmenter le nombre de sorties ;
- La valeur obtenue par le capteur de son doit se trouver sur une échelle adaptée (si le son est fort, beaucoup de LEDS doivent s'allumer)

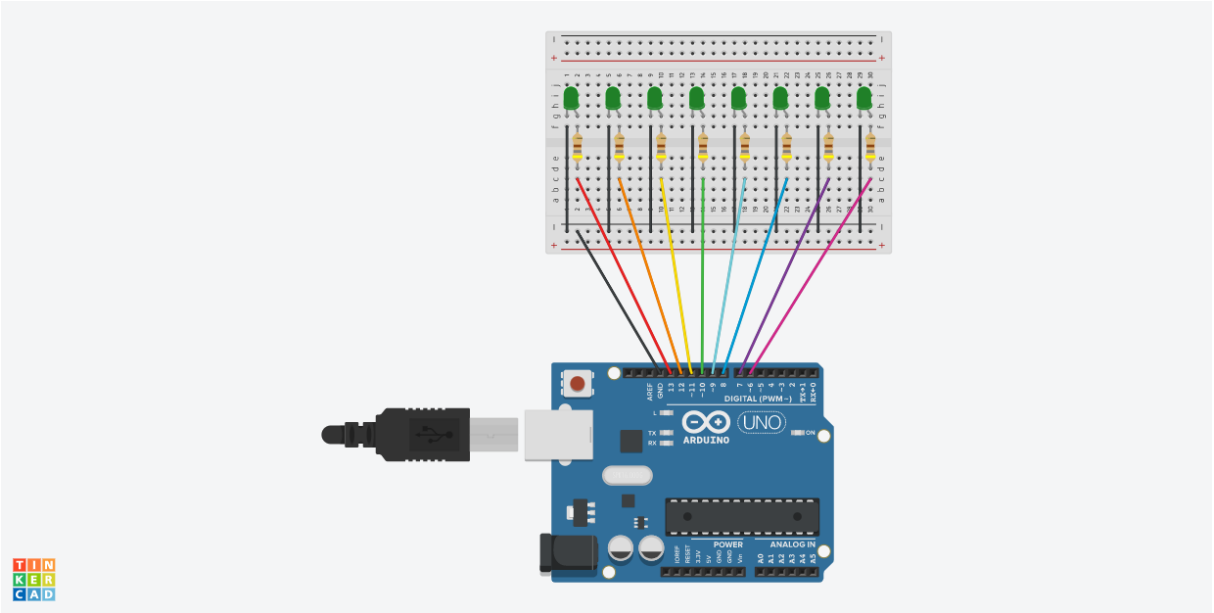



*Figure 6 : Le nombre de LEDs 8 - ajout des résistances*

La première modification (lien) apportée à ce programme est la modification du « else if » en « if » qui permet d'allumer toutes les LEDS qui sont inférieurs à la valeur analogique du capteur de son.

Vous retrouverez toutes les modifications de programme en annexe.

- Dans l'exemple 2 (lien), nous avons modifié les délais qui permettent aux LEDS de s'éteindre une par une (OK). Nous voulions améliorer le visuel et le design, cependant la différence de temps entre le son perçu et la visualisation via les LEDS est trop long. (Echec)
- Durant la troisième modification (lien), nous avons ajouté 4 LEDS supplémentaires et donc par ailleurs changé l'échelle pour allumer les LEDS. (OK)
- La quatrième modification (lien) comporte un compteur pour éviter d'entrer dans le programme si la valeur analogique est de zéro. Finalement, nous remarquons qu'il n'est pas nécessaire d'ajouter un compteur, mais juste d'ajouter une condition comme (if valeur > 5). (OK)
- Dans la prochaine modification (lien), nous avons adapté l'échelle du capteur analogique et nous avons aussi modifié les conditions pour allumer les LEDS. Les conditions ne sont plus linéaires comme par exemple (1 ;2 ;3 ;4 ;5…) mais bien exponentielles (1 ;2 ;3 ;5 ;8 ;13…) puisque le son est exponentiel. Les effets visuels sont plus agréables à observer. (OK)
- Le sixième programme (lien) comporte les conditions exponentielles sans la modification du signal analogique. (Echec)
- Le dernier programme (lien) comporte une modification du signal adapté aux conditions. (OK) [7]

Diagramme de fonctionnement :

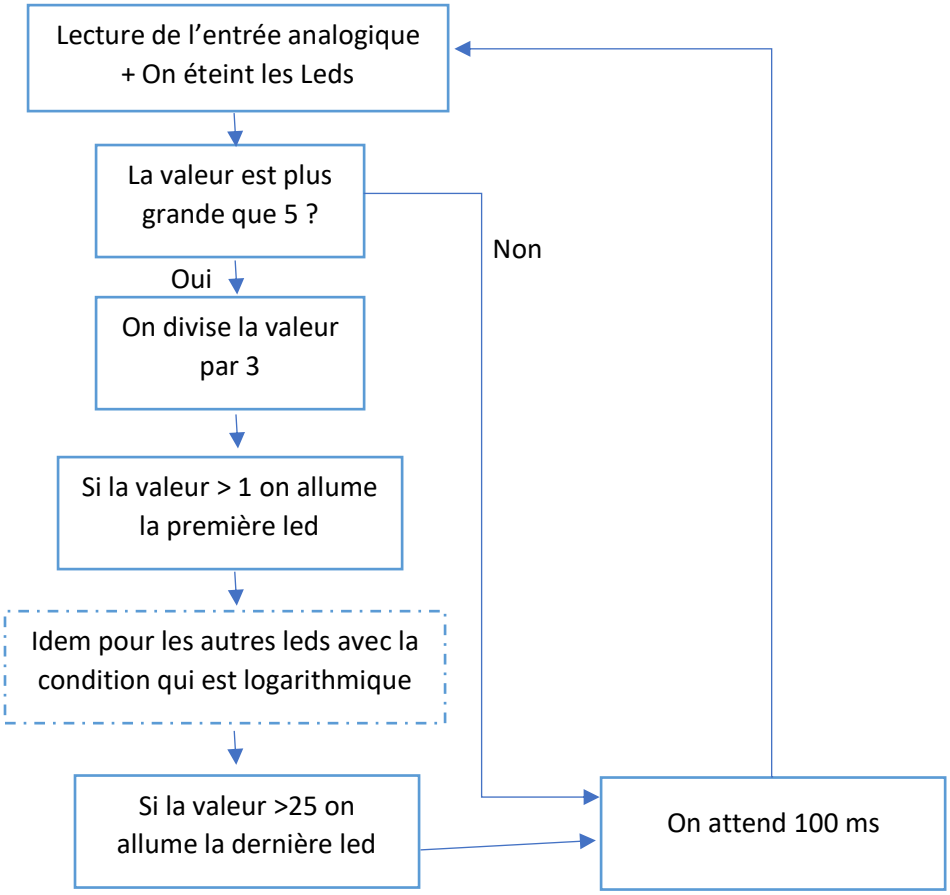



*Figure 7 : Diagramme fonctionnement du dernier programme Arduino*

# III. Raspberry pi 1

## 1. Fonctionnement de l'algorithme de reconnaissance

Pour mener à bien notre projet, nous avons fait appel à un algorithme de reconnaissance qui permet de faciliter certaines tâches comme faciliter la communication, ou alors optimiser l'espace, pas de bouton physique mais que des gestes pour commander le jeu par exemple. Il est possible de réaliser la reconnaissance de geste, comptabiliser le nombre de doigt, les mouvements de la mains,… Pour des questions de simplicité, nous avons commencer la réalisation de ce programme avec un traitement d'image pour comptabiliser le nombre de doigts.

Tout d'abord, le logiciel utilise la méthode de « background subtraction ». Cela veut dire que le programme connait le « fond » de l'image et il soustrait ce fond à l'image qu'il traite. De cette manière, l'algorithme extrait les endroits de l'image ou il se passe quelque chose de spécial. (Par exemple les voitures en blanc)

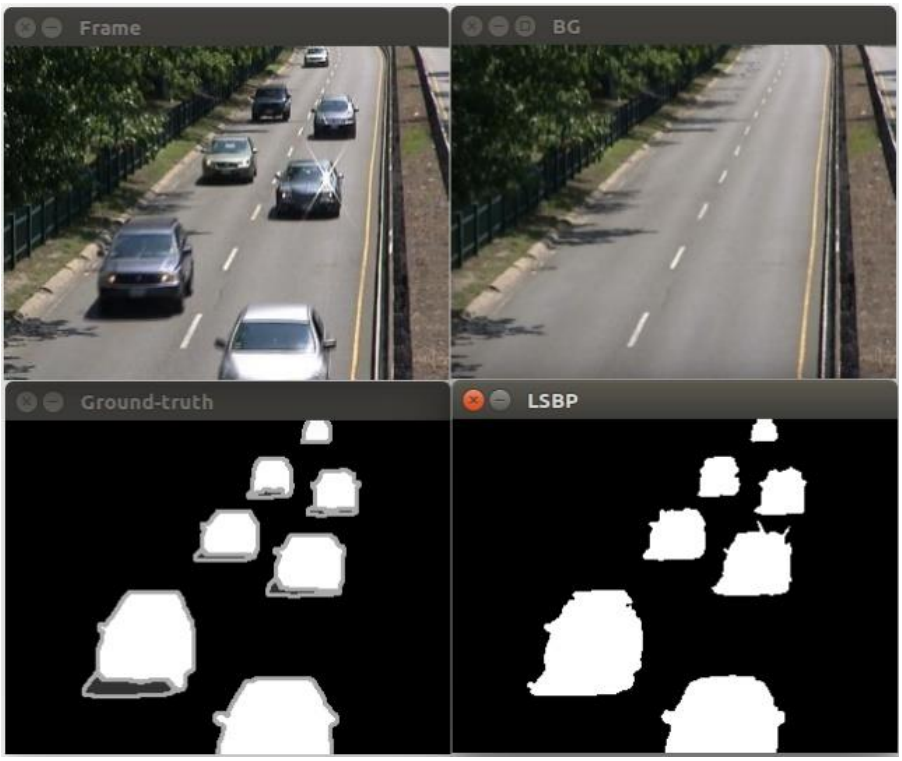



*Figure 8 : Exemple de frames*

Pour « connaitre » le fond, l'algorithme que l'on utilise se calibre automatiquement au démarrage. Il va enregistrer les 30 premières frames qu'il va considérer comme le background. [8]

Pour compter le nombre de doigts, la reconnaissance se fait en 5 étapes. [9]

1) Avec l'image de la main que l'on a extraite plus tôt, on cherche les coordonnées des points aux 4 extrémités (en haut, à gauche, à droite, en bas)

2) On fait une moyenne de ces points pour définir le centre de la main

3) On construit un cercle autour du point central. Le rayon est la distance euclidienne entre ce centre et les points extrêmes

4) On fait un ET entre la première image de la main et le cercle que l'on vient de construire. On obtient un cercle « coupé »

5) Il suffit de compter le nombre de coupure -1 de ce cercle et on à le nombre de doigts.


*Figure 9 : Fonctionnement du programme*

On a choisi de faire fonctionner le processus dans le coin supérieur droit du champ de vision de la caméra.

Limiter la zone de reconnaissance à une partie de l'image permet de ne cibler que la main et donc avoir des meilleurs résultats. En effet, vu le fonctionnement de la reconnaissance il est impératif de n'avoir que la main dans cette zone. Une partie de l'épaule ou un bras trop long peut facilement perturber la mesure. [10]

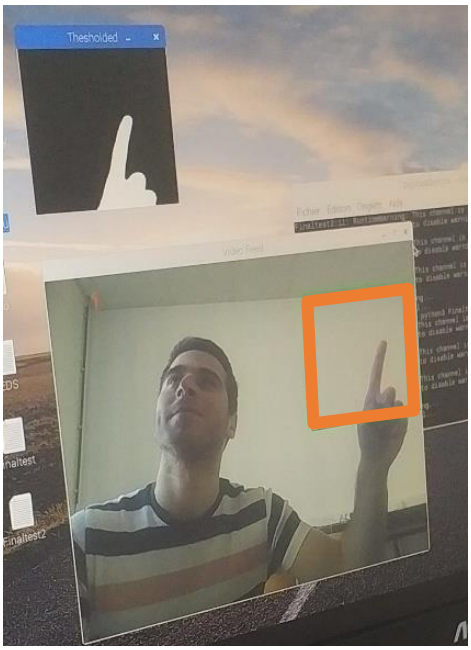



*Figure 10 : Exemple de fonctionnement - Traitement de l'image*

## 2. Critique de l'approche

### a) Le code existant

Le code que l'on utilise provient du GitHub suivant : https://github.com/Gogul09/gesture-recognition

Ce code est accompagné d'explications et est même divisé en plusieurs partie. Cela permet de pouvoir tester les parties indépendamment l'une de l'autre et les corriger.

Corrections apportées :

- Chercher partout ou des « tab » ont été utilisée et les remplacer par des « espaces »
- Rajouter des parenthèses à la fonction « print » (à cause de la version de python)
- Remplacer « (-, cnts, -) » par « (cnts, -) » (à cause de la version d'OpenCV)

Optimisation appliquée :

- Modification des valeurs de la position du rectangle pour la prise de mesure (ligne 115)
  Top = 65 / Bottom = 460 / Left = 280 / Right = 700

### b) Analyse

Comme expliqué plus tôt, l'algorithme traite l'image et donc ne fait pas appel au machine learning. Ceci à l'avantage d'être plus simple à mettre en œuvre (pas de phase d'apprentissage, etc ...) mais en contrepartie on n'obtient pas les mêmes résultats.

En effet, le machine learning, permet d'apprendre à la machine des gestes que l'on définit nous-mêmes, or ici ce n'est malheureusement pas le cas. L'algorithme n'est calibré que pour compter le nombre de doigts.

Le Code que nous avons utilisé fonctionne, mais il est très sensible aux perturbations. En effet, à la moindre modification de l'environnement (une simple variation de luminosité par exemple) le fonctionnement est complétement perturbé et il faut recommencer le calibrage.

(Pour un fonctionnement optimal, un fond blanc est idéal.)

Sinon, de manière générale, le programme est bien structuré et bien commenté. Cela nous a permis d'y intégrer facilement notre code pour la commande des pin GPIO en fonction du nombre de doigts détectés. (expliqué dans le paragraphe suivant).

Grâce à la lecture et à l'analyse du code, nous avons pu grandir en connaissance, notamment sur le travail de l'image via un algorithme.

## c) Fonctionnement de la reconnaissance des doigts

Dans l'algorithme de la reconnaissance de doigts, nous avons ajouté du code pour pouvoir interagir avec les pins GPIO selon le nombre de doigts que le logiciel reconnait.

Voici un diagramme du fonctionnement de ce code [11] :

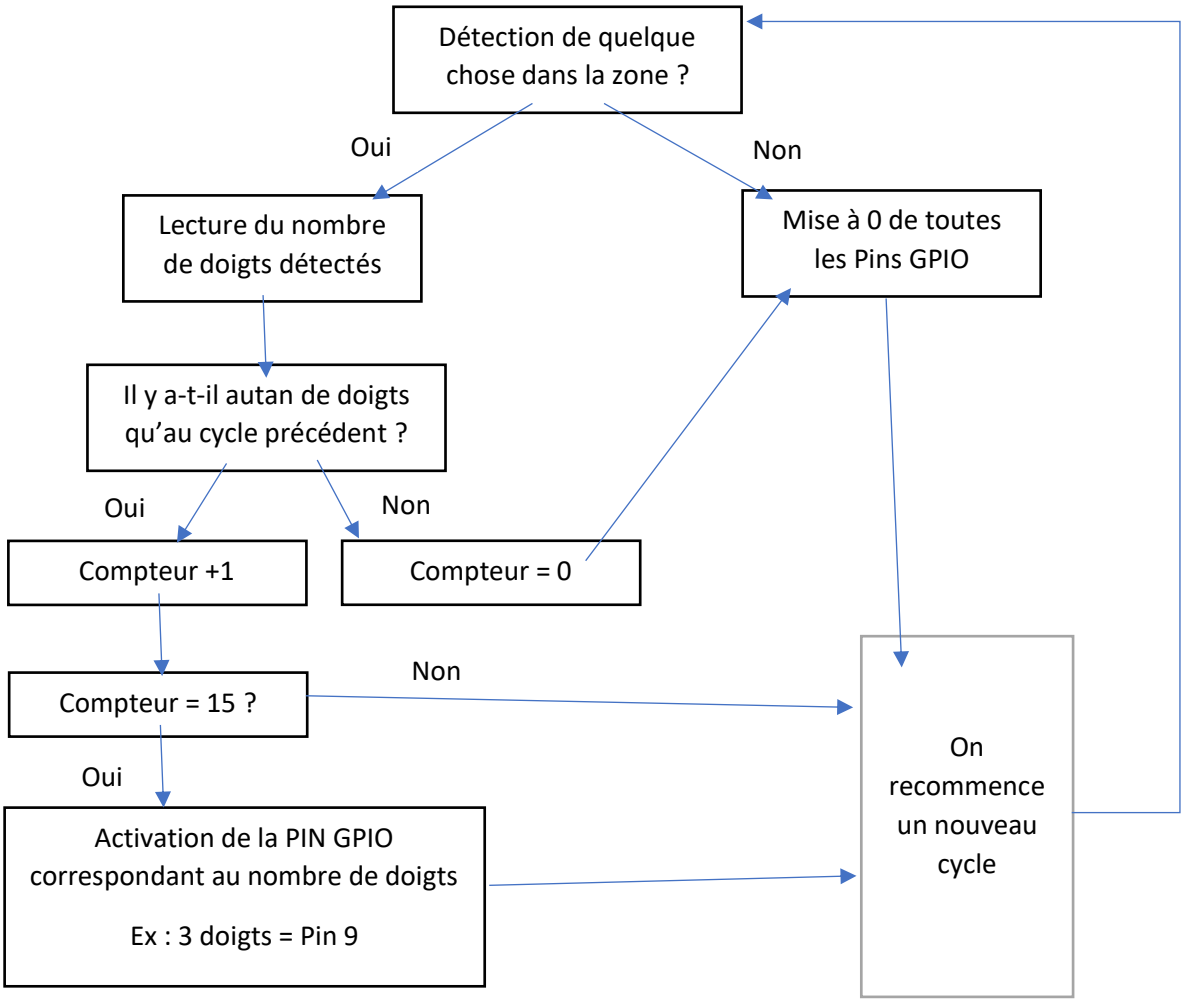



*Figure 11 : Diagramme de fonctionnement*

Remarque :

Nous avons choisi une valeur de 15 pour le compteur car c'est un compromis entre rapidité et fiabilité :

Avec un compteur plus petit, la reconnaissance est plus rapide mais le risque que la machine pense voir quelque chose de faux est plus élevé.

A l'inverse, si on augmente ce nombre il n'y quasi plus de risque d'erreur mais le temps de reconnaissance devient très long.

## 3. Transmission de l'information du Raspberry 1 au Raspberry 2

Comme expliqué dans le diagramme précédent, la détection d'un certain nombre de doigts active certaines pin GPIO du Raspberry 1.

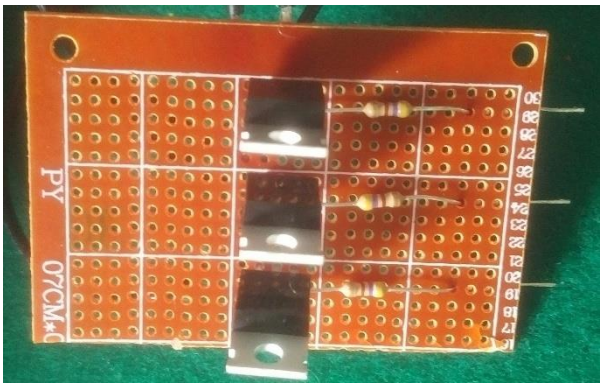
Nous avons choisi cette solution car il faut passer par une méthode de communication hardware.

En effet, le Raspberry pi 2 est formaté avec un système d'exploitation spécifique pour les jeux vidéo, il est donc très difficile de communiquer avec lui via des protocoles de communication (spi, i2c, …).

Nous avons donc décidé de simuler un appui sur les boutons physiques en plaçant un transistor mosfet en parallèle avec les boutons poussoirs. [12 - 14]

Malheureusement, la tension de sortie d'un pin GPIO n'est que de 3,3v et nous ne savons pas piloter le mosfet directement. C'est pourquoi nous passons par l'Arduino qui va transformer le signal 3,3V en un signal 5v.

*Figure 12 : Câblage des transistors*



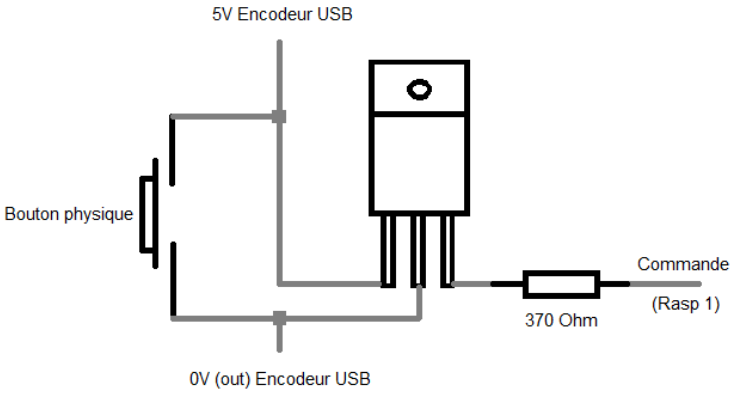
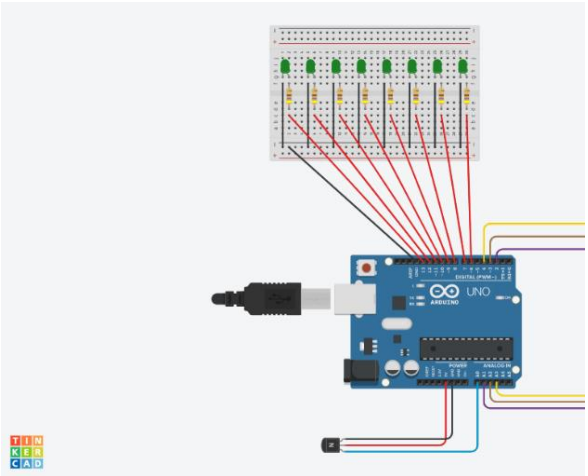



*Figure 13 : Schéma de câblage et image de la plaque électronique*



Fils rouges Leds – Sortie digitale Arduino

Fils noirs – Ground

Jaune/Marron/Mauve – Sortie digitale Arduino vers Raspberry 1

Rouge (5V) – Noir (Ground) – Bleu (entrée analogique)

Jaune/Marron/Mauve – Sortie analogique Arduino vers Raspberry 2

*Figure 14 : Schéma de câblage Arduino + Raspberry 2*

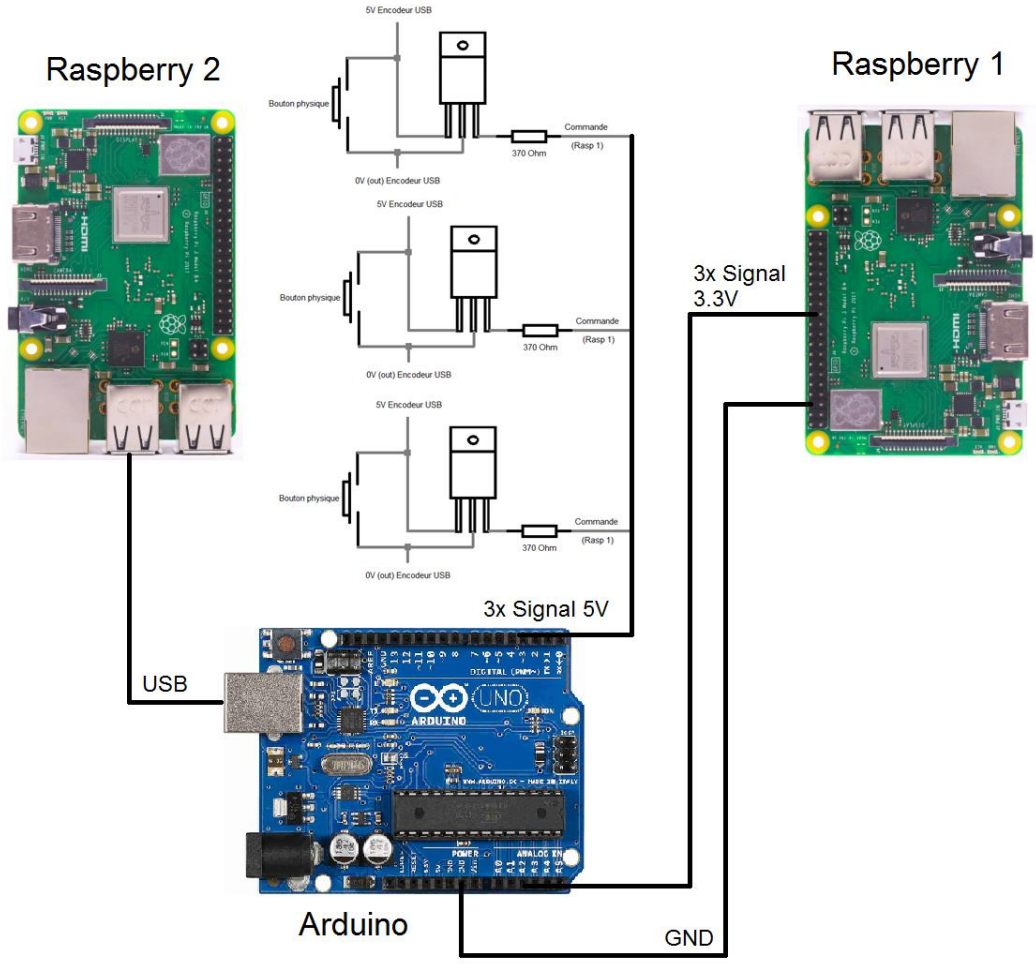



*Figure 15 : Diagramme récapitulatif*

## 4. Installation des librairies

Hormis l'installation d'OpenCv sur Raspberry, il est facile d'installer les différentes librairies avec la fonction « pip install <Nom de la librairie> » dans l'invite de commande.

Pour notre algorithme de reconnaissance d'extrémité de la main, il faut installer les librairies suivantes :

- Imutils (outil permettant le traitement d'image)
- Numpy (outil de calculs matriciels avancés)
- Sklearn (outil pour le machine-learning)

Pour l'installation d'OpenCv sur Raspberry, nous vous conseillons de suivre le tutoriel suivant :

https://www.pyimagesearch.com/2018/09/26/install-opencv-4-on-your-raspberry-pi/

Note : nous vous conseillons également d'utiliser une carte de 16Go min afin de ne pas être saturé en mémoire.

# IV. Raspberry pi 2 : Installation de Retropie

La première étape est de télécharger le système d'exploitation sur le site de Retropie. [16]

Cependant, le système seul ne permet pas de jouer.

En effet, Retropie va « simuler » des anciennes consoles à partir de la puissance de calcul du Raspberry. (Le fonctionnement est identique à une machine virtuelle sur pc)

Du coup il faut également télécharger des jeux que l'on trouvait sur ces consoles. Nous installerons ces derniers plus tard.

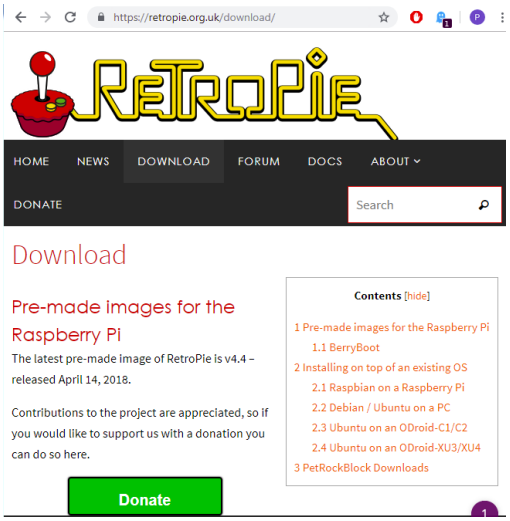



*Figure 16 : Page d'accueil du site web de Retropie*

Ensuite, il faut installer l'image de Retropie sur la carte micro-sd du Raspberry. Pour ce faire il faut tout d'abord formatter cette carte à l'aide du logiciel « sd card formatter » par exemple. [17]

Après le formatage on peut installer l'image à l'aide de « Win32DiskImager ». [18]

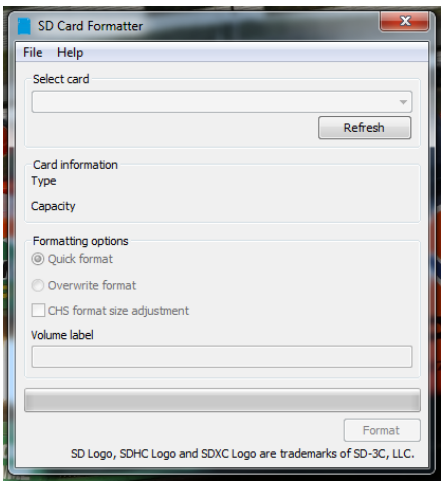



*Figure 17 : Formatage de la carte SD*

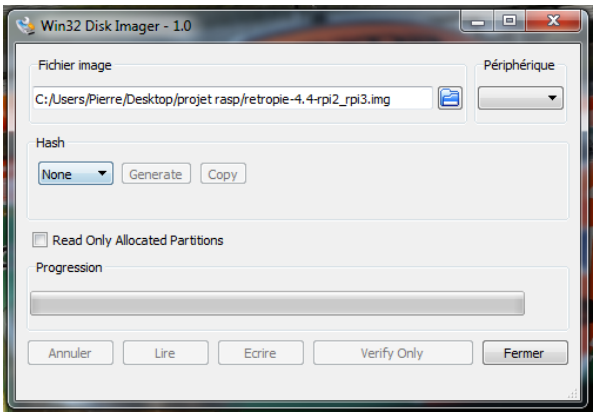



*Figure 18 : Win32 Diskimager*

A partir de maintenant on peut remettre la carte dans le Raspberry et passer à la configuration des touches.

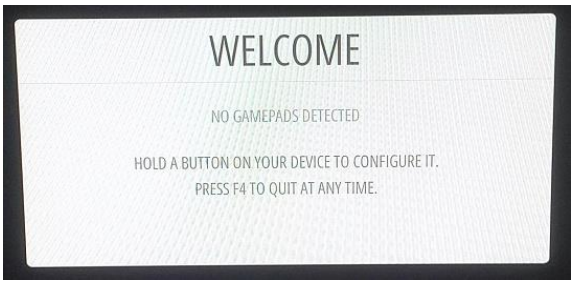



*Figure 19 : Ecran d'acceuil du système d'exploitation Retropie*

Pour enregistrer les touches, nous avons raccordé et étiqueté tous les boutons ainsi que les joysticks sur l'encodeur usb.

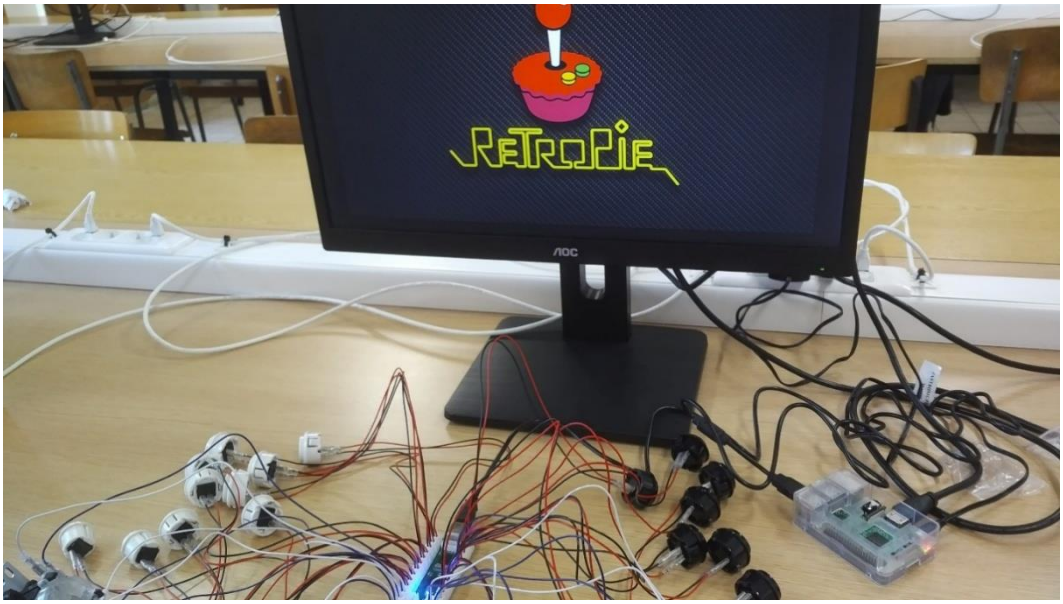



*Figure 20 : Premier démarrage de la console Retropie (Raspberry 1)*

Cependant étant donné que nous avons 2 jeux de boutons (car prévu pour 2 joueurs), il va falloir modifier le materiel pour que l'encodeur usb soit considéré comme deux « manettes » différentes et non une seule.

Voici la démarche qu'il à fallu réaliser :

1) Dans les menus de retropie, executer file manager
2) Se déplacer dans le répertoire : → /pi → /.. → /boot
3) Editer le fichier « CMDLine.txt » avec la touche « F4 » puis appuyer sur « 2 »
4) A la fin du fichier, rajouter le code suivant : « USBhid.quirks=0x0810 :0xe001 :0x00000040 »
5) Valider en appuyant sur les touches : « CTRL+0 » → « enter » → « F10 »

Voici le plan de raccordement des touches sur l'encodeur [21] :

| **Joueur blanc** | | | **Joueur noir** |
|---|---|---|---|
| Right trigger | 1 | 15 | Right trigger |
| Left trigger | 2 | 16 | Left trigger |
| Right shoulder | 3 | 17 | Right shoulder |
| Left shoulder | 4 | 18 | Left shoulder |
| Y | 5 | 19 | Y |
| X | 6 | 20 | X |
| B | 7 | 21 | B |
| A | 8 | 22 | A |
| Select | 9 | 23 | Select |
| Start | 10 | 24 | Start |
| UP (joystick) | 11 | 25 | UP (joystick) |
| Down (joystick) | 12 | 26 | Down (joystick) |
| Left (joystick) | 13 | 27 | Left (joystick) |
| Right (joystick) | 14 | 28 | Right (joystick) |

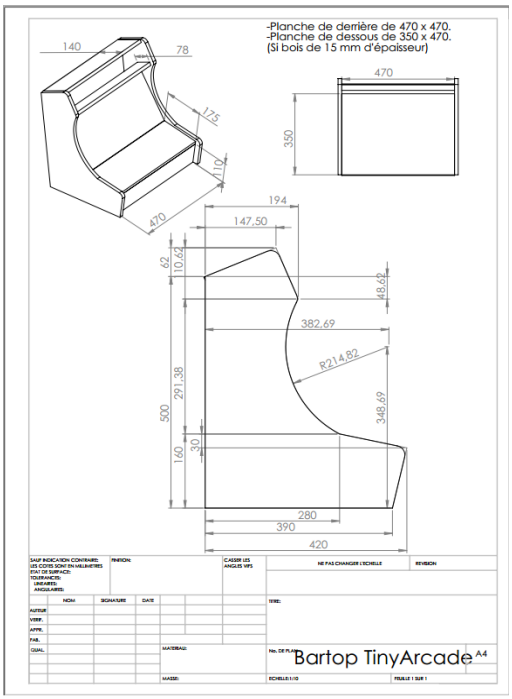
*Figure 21 : Plan de raccordement des boutons*

# V. Réalisation de la borne.

Pour réaliser notre borne, nous avons fait découper le bois par les secondaires de Pierrard.

Nous avons récupéré un plan qui provient d'internet et nous l'avons corrigé par rapport aux dimensions de l'écran que l'on compte utiliser.

Ensuite, nous avons percés des trous pour les boutons. Nous avons choisi le design de deux rangées de 4 boutons car de cette manière tous les boutons sont accessibles depuis la main droite. [22]



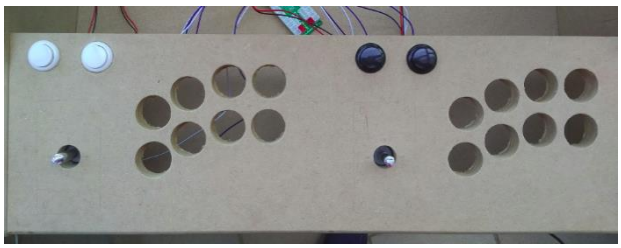



*Figure 22 : Disposition des boutons et plan de la borne arcade*

Il reste alors à assembler le tout et de brancher les boutons sur le bornier pour avoir la structure de notre borne. [23]

*Figure 23 : Avancement général*

# VI. Conclusion

En conclusion, nous sommes parvenus à obtenir quelque chose de fonctionnel mais cela n'aura pas été facile.

Tout d'abord nous avons eu quelques difficultés pour faire communiquer les deux Raspberry ensemble (on a dû passer par l'intermédiaire de l'Arduino).

Mais la plus grande difficulté à surtout été de trouver et de faire fonctionner un algorithme de reconnaissance provenant d'internet. Nous avons fait de très nombreux essais infructueux et fait face à toute sorte de problèmes (compatibilité python-OpenCv, librairies difficiles à installer, passage du pc au Raspberry, carte SD saturé, etc …).

Fort heureusement, on a pu résoudre tout ces problèmes et pu tout combiner pour avoir notre borne d'arcade fonctionnelle.

Il y a cependant des points à améliorer, notamment la reconnaissance. Il serait fort intéressant de faire un algorithme qui utilise le machine learning, on pourrait alors diminuer fortement le temps de reconnaissance et limiter les problèmes de perturbation grâce à l'apprentissage de gestes spécifiques.

Enfin, nous avons trouvés très intéressant d'avoir pu approcher les concepts de traitement d'image et d'intelligence artificiel dans notre parcours d'étude.

# VII. Bibliographie


https://www.pyimagesearch.com/2018/09/26/install-opencv-4-on-your-raspberry-pi/

https://github.com/iftheqhar/opencv2_python/blob/master/software/firmware/cam.py

https://github.com/lzane/Fingers-Detection-using-OpenCV-and-Python

https://github.com/mahaveerverma/hand-gesture-recognition-opencv

https://www.framboise314.fr/i-a-realisez-un-systeme-de-reconnaissance-dobjets-avec-raspberry-pi/

https://github.com/SaiVinay007/hand-gesture-recognition

Code de base de notre projet :

https://gogul09.github.io/software/hand-gesture-recognition-p1

https://gogul09.github.io/software/hand-gesture-recognition-p2

Notre lien Github :

https://github.com/DemirYasin/ArcadeCube

# Table des figures